2011

Large Scale Distributed Semantic N-gram Language Model

Yuandong Jiang
Wright State University

Large Scale Distributed Semantic N-gram Language Model

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Engineering

By

Yuandong Jiang

B.E.,Dalian Jiaotong University 2009

2011

Wright State University

WRIGHT STATE UNIVERSITY

GRADUATE SCHOOL

August 12,2011

I HEREBY RCOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Yuandong Jiang ENTITLED Large Scale Distributed Semantic N-gram Language Model BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Computer Engineering.

_____

Shaojun Wang, Ph.D.

Thesis Director

_____

Mateen Rizki, Ph.D., Chair

Department of Computer Science and Engineering

Committee on

Final Examination

_____

Shaojun Wang, Ph.D.

_____

Keke Chen, Ph.D.

_____

Xinhui Zhang, Ph.D.

_____

Andrew Hsu, Ph.D.

Dean, Graduate School

Abstract

Yuandong Jiang. M.S.C.E., Department of Computer Science and Engineering, Wright State University, 2011. *Large Scale Distributed Semantic N-gram Language Model.*

Language model is a crucial component in statistical machine translation system. The basic language model is N-gram which predicts the next word based on previous N-1 words. It has been used in the state-of-the-art commercial machine translation systems over years. However, the N-gram model ignores the rich syntactic and semantic structure in natural languages. We propose a composite semantic N-gram language model which combines probabilistic latent semantic analysis model with N-gram as a generative model. We have implemented the proposed composite language model in a super-computer with thousand processors that is trained by 1.3 billion tokens corpus. Comparing with simple N-gram, the large scale composite language model has achieved significant perplexity reduction and BLEU score improvement in an n-best list re-ranking task for machine translation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The goal of statistical language modeling is to accurately model the probability of naturally occurring word sequences in human natural language, and it is crucial for the success of a wide spectrum of robust and intelligent language technology applications. These include speech recognition (Jelinek 1998) (from which the statistical language model originated), machine translation (Brown 1993), information retrieval (Croft and Lafferty 2003, Ponte and Croft 1998), disfluency modeling (Johnson et al. 2004), sentence compression (Turner et al. 2005), and many more. The first and simplest language model is the Markov chain (N-gram) source models, which predicts each word on the basis of previous N-1 words. It was first explored by Shannon in his seminal paper (Shannon 1948) to illustrate many features of information theory. In the early 1970s, the N-gram together with the hidden Markov model for acoustic signals was applied to automatic speech recognition under the source-channel paradigm by Jelinek and his colleagues at IBM and achieved a dramatic improvement in speech recognizer performance. In the late 1980s, the N-gram models were used by the same group at IBM for machine translation in the same way as in speech, but became just as important for obtaining good performance. Subsequently, a wide variety of smoothing methods (Chen and Goodman 1999) have been developed to address the

problem of estimating rare events for these models. The resulting smoothed N-gram language models have been the workhorses of the state-of-the-art of speech recognizers and machine translators, which help to resolve acoustic or foreign language ambiguities by placing higher probability on more likely original underlying word strings. In 1998, Ponte and Croft (Ponte and Croft 1998) used a smoothed unigram language model for information retrieval, and this simple language model was remarkably effective ``right out of the box." Most recently, Durbin's Genome Bioinformatics group (Coin et al. 2003) showed that the N-gram models can significantly enhance domain recognition in protein sequences.

Although the Markov chains are efficient at encoding local word interactions, the N-gram model clearly ignores the rich syntactic and semantic structure that constrains natural languages. Attempting to increase the order of an N-gram to capture longer range dependencies in natural language immediately it runs into the curse of dimensionality (Bengio et al. 2003). Consider, for instance, predicting the word *fell* from the word *stocks* in the two equivalent phrases (Bellegarda 2000):

stocks fell sharply as a result of the announcement

and

stocks, as a result of the announcement, sharply fell

In the first phrase, the prediction can be done with the help of a bigram language model. In the second phrase, however, the value n=9 would be necessary, a rather unrealistic requirement. In large part because of this inability to reliably capture large-span behavior, the performance of conventional N-gram technology has essentially reached a plateau (Rosenfeld 2000b), and it has proven remarkably difficult to improve on N-grams (Jelinek 1991).

Only within the last decade have more sophisticated models been developed that significantly outperform N-grams; these are mainly the syntactic language models

(Charniak 2001, Chelba and Jelinek 2000, Roark 2001) that effectively exploit the syntactic structure of natural language, and the topic language models (Bellegarda 2000, Hofmann 2001, Saul and Pereira 97) that exploit document-level semantic content. In the syntactic language model, each unit is in the form of the headword of the phrase spanned by the associated parse subtree. The syntactic language model operates given the last headwords as opposed to the last words. In the "stocks-fell" example, the top two headwords in the dependency graph would be *stocks* and *fell* in both cases, thereby solving the problem. Impressive improved performance has been achieved by using the syntactic models instead of N-grams. For example, in traditional language model applications, Chelba and Jelinek (Chelba and Jelinek 2000) obtained more than 1% absolute word error rate reduction on the notoriously difficult Switchboard corpus in speech recognition. Charniak et al.'s research (Charniak et al. 2003) in machine translation found that by moving from an N-gram to a syntactic language model, human judges deemed 50% more of translations to be perfect, 200% more to be grammatically correct, but an equivalent number to semantically correct. In novel language model applications, Johnson and Charniak's syntactic model yielded a 3% f-measure improvement (12.5% error reduction) over an N-gram model in disfluency modeling (Johnson and Charniak 2004), and 10% more compression (i.e. shorter paraphrases) with no loss in grammaticality or information in sentence compression (Turner et al. 2005). In the topic language model (Bellegarda 2000, Hofmann 2001, Saul and Pereira 1997), the fundamental idea is to discover compact semantic representation of text data that determines the similarity of the meaning of words and passages to each other. In the "stocks-fell" example, the topic model reveals a significant correlation between *stocks* and *fell* irrelevant to the distance between *stocks* and *fell*. For the second phrase, when the topic model is integrated with an N-gram model to form a semantic N-gram model, the presence of *stocks* in the document could automatically trigger *fell*, causing its conditional probability estimate, given its history, to increase, thus the semantic N-gram model enables the

probabilities of these two phrases to be at the same order. In fact, a substantial word error rate reduction, 16%, over a trigram baseline in speech recognition task (Bellegarda 2000), is obtained. All the results above were obtained several years ago by training the models on a corpus of millions of words. However, conceptually, as long as the models are estimated by EM algorithms (Dempster et al. 1977), they can be scaled up to billions or trillions of words via distributed computing, where the E step is computed for each sentence of every document and thus can be performed by a local worker, and the M step can be excuted at the central server. It is reasonable to expect that these models will have similar improvements over N-grams when trained on up billions or trillions of words.

Unfortunately, each of these language models described above only targets some specific, distinct linguistic phenomena (Pereira 2000, Rosenfeld 2000), thus each captures and exploits different aspects of natural language regularity. Previous techniques for combining language models are either unviable due to unrealistic assumptions to be effective, i.e., linear additive form in linear interpolation (Jelinek and Mercer 1981), or due to an intractable model assumption, i.e., undirected Markov random fields (Gibbs distributions) in maximum entropy (Mark et al. 1996, Rosenfeld 2000).

In this thesis, we propose a composite semantic N-gram language model which combines probabilistic latent semantic analysis model with N-gram as a generative model. We have implemented the proposed composite language model in a super-computer with thousand processors that is trained by 1.3 billion tokens corpus. Comparing with simple N-gram, the large scale composite language model has achieved significant perplexity reduction and BLEU score improvement in an n-best list re-ranking task for machine translation.

In Chapter 2, we first introduce the well-known source-channel model that lays down the foundation of statistical machine translation. Then we describe the basic N-gram language model and its linear smoothing technique. We present PLSA model in section 2.3, and the

concept of perplexity as a measure of the goodness of language model in section 2.4. We present our proposed composite semantic N-gram language model in Chapter 3, where PLSA is used to model the long range semantic content at document level and N-gram is used to model local word interaction. We derive EM algorithm to train this composite model in a distributed fashion to handle large scale corpora. In Chapter 4, we present the experimental results on the giga English corpus with billion words, where we obtain significant perplexity reduction and BLEU score improvement in an n-best list re-ranking task for machine translation. We draw our conclusion and point out future work in Chapter 5.

Chapter 2

Background

2.1 Source-Channel Model

The state-of-the-art machine translation system is based on the so-called source-channel model that was first proposed by IBM researchers (Brown et al. 1988) in late 1980s.

Let's formalize the source-channel model for statistical machine translation. Given a sentence F in the target language, the task of machine translation is to search the sentence E from which the translator produced F. The chance of making errors would be minimized by choosing that sentence E that is most likely given target sentence F. Thus, we would like choose so as to maximize $P(E|F)$. Using Bayes rule, we have:

$$P(E|F) = \frac{P(F|E)P(E)}{P(F)} \quad (1)$$

The denominator on the right of the above equation does not depend on E, so it is sufficient to choose E that maximizes the product of $P(F|E)$ and $P(E)$. We call the first factor in the product, $P(F|E)$, *translation model* and the second factor $P(E)$ as *language model.*

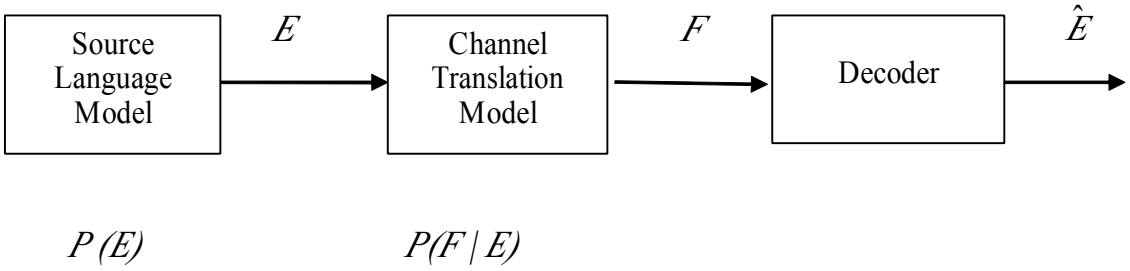

Figure 1 *A source language model* and a channel *translation model* describe a joint probability distribution over source-target sentence pairs (E, F). The joint probability P(E, F) is the product of probability P(F) given by the language model and the conditional probability P(F | E) given by the translation model. The parameters of these models are learned estimated from a corpus of source sentences as well as a corpus of source-target sentence pairs. A *decoder* performs the actual translation.

Thus as illustrated in Figure 1, a statistical machine translation system requires a method to compute the probabilities of translation model, a method to compute the probabilities of language model, and finally a method for searching the source sentence $\hat{E}$ that gives the highest value of P (E | F) P (E) among all possible source sentences.

$$\hat{E} = \arg\max_{E} P(F \mid E) P(E) \tag{2}$$

This thesis focuses on a method to accurately compute the probabilities of the language model that takes into account document level semantic content, and is scalable to be trained by large scale corpora.

2.2 N-gram

The basic language model is N-gram. Denote word string $w_1,...,w_n$ as W_1^n. By chain rule, the joint probability of a sequence string W_1^n is the product of the conditional probability of a word given with its history, that is,

$$P(W_1^n) = P(w_1)\,P(w_2 \mid w_1)\,P(w_3 \mid w_1^2)\ldots P(w_n \mid w_1^{n-1})$$

$$= \prod_{k=1}^{1} P(w_k \mid w_1^{k-1})$$

With the assumption that when we generate word w_k, it only depends on its previous N-1 words but not beyond, that is,

$$P(w_k \mid w_1^{n-1}) \approx P(w_k \mid w_{n-N+1}^{n-1})$$

Thus the probability of a completed word sequence can be computed by the following approximation:

$$P(w_1^n) \approx \prod_{k=1}^{1} P(w_k \mid w_{n-N+1}^{n-1})$$

We can use maximum likelihood estimation (MLE) to estimate $P(w_k \mid w_{n-N+1}^{n-1})$ from a training a corpus, which leads to a relative frequency counts, $\#(w_k\ w_{n-N+1}^{n-1})/\#(w_k\ w_{n-N+1}^{n-1})$.

To overcome overfitting problem, various smoothing techniques have been proposed. The simplest smoothing technique is linear interpolation, where the probability of N-grams is a linear combination of lower orders. For example, consider a trigram, we can compute $P(w_n \mid w_{n-1} w_{n-2})$ by combining unigram, bigram and trigram probabilities together with weighted λ:

$$\hat{P}(w_n \mid w_{n-1} w_{n-2}) = \lambda_1 P(w_n \mid w_{n-1} w_{n-2})$$

$$+\lambda_2(w_n|w_{n-1})$$

$$+\lambda_3 P(w_n)$$

And the sum of λ should be 1:

$$\sum_i \lambda_i = 1$$

In a little bit complex perspective of linear interpolation, it is a complicated way to calculate λ weight conditionally. Actually we can make the trigram more weight by higher λs of those trigrams. We suppose it is trustworthy for those counts of trigrams which rely on the bigram, what if the count for a bigram is exactly correct.

Hence, we have interpolation with context-conditioned weights:

$$\hat{P}(w_n|w_{n-1}w_{n-2}) = \lambda_1(w_{n-2}^{n-1})P(w_n|w_{n-2}w_{n-1})$$

$$+\lambda_2(w_{n-2}^{n-1})P(w_n|w_{n-1})$$

$$+\lambda_3(w_{n-2}^{n-1})P(w_n)$$

We should understand how to adjust the value of λ. As far as we know, the λ is drove from held-out corpus which is a training corpus we use to calculate parameters in both simple and conditional interpolations. In this way, we can get λ from those values and the value of λ normally maximizes the likelihood of corpus.

We check the value of λ after the N-gram probabilities are stabilized, and then get the most probability of the held-out set. Regularly, we think about EM algorithm to detect the optimal value of λ.

2.3 PLSA

A document can be viewed as a collection of semantically homogeneous sentences. Given a large number of documents, latent semantic analysis (LSA) (Bellegarda 2000, Deerwester et al. 1990) attempts to discover compact semantic representations of text data that go beyond simple lexical-level word co-occurrences. This is achieved by mapping a high-dimensional vector representation of documents (term-frequency vectors) to a lower dimensional representation in a so-called latent semantic space. Semantic relations between words and documents can then be easily defined in terms of their proximity in the semantic space by dimensionality reduction techniques (Bellegarda 2000, Hofmann 2001). Recently, much advanced topic models have been developed (Blei et al. 2003, Blei et al. 2005b, Blei and Lafferty 2006). It has been shown that exploiting latent semantic information can significantly improve language models for automatic speech recognition versus N-gram models (Bellegarda 2000).

Probabilistic latent semantic analysis (PLSA), also known as probabilistic latent semantic indexing (PLSI, especially in information retrieval community) is a statistical technique for the analysis of word document co-occurrence data. Standard latent semantic analysis (LSA) stems from linear algebra and downsizes the occurrence tables via a singular value decomposition, PLSA is evolved from latent LSA and it is based on a mixture decomposition derived from a latent class model. This results in a more principled approach which has a solid foundation in statistics. Since its invention by Thomas Hofmann in 1999 (Hofmann 1999), it has been widely applied in information retrieval and filtering, natural language processing, machine learning from text, computer vision and many related areas.

Assume we have a training corpus which is a collection of documents $D = \{d_1, d_2, ..., d_L\}$. Assume the vocabulary is $V = \{w_1, w_2, ..., w_M\}$. PLSA is a generative probabilistic model for word-document co-occurences that makes the bag-of-words assumption and can be

described by the following processes:

Choose a document d with probability P (d_i);

Select a topic z_k have probability of P (z_k | d_i);

Pick a word w_j from the topic z_k with probability of P (w_j | z_k).

For all these occurrence with a pair of (d_i,w_j), the latent topic of variable z_k is unobserved. So we will have the expression for data generation in a probability model:

$$P(d_i, w_j) = P(d_i)\, P(w_j | d_i)$$

$$P(w_j | d_i) = \sum_{k=1}^{K} P(w_j | z_k) P(z_k | d_i)$$

Our goal is to learn the unknown parameters P(d), P(w | z), P(w | z). We use maximum likelihood formulation to estimate these unknown parameters through maximizing the following log-likelihood over training corpus D:

$$\mathcal{L} = \sum_{i=1}^{L} \sum_{j=1}^{M} n(d_i, w_j) \log P(d_i, w_j)$$

$$= \sum_{i=1}^{L} n(d_i) \left[\log P(d_i) + \sum_{j=1}^{M} n(d_i, w_j) / n(d_i) \log \sum_{k=1}^{K} P(w_j | z_k) P(z_k | d_i) \right]$$

In this function, $n(d_i) = \sum_j n(d_i, w_j)$ means length of a document.

In latent variable model, the most widely used method for maximum likelihood estimation is the so-called Expectation Maximization (EM) algorithm. There are two steps in EM. One is the Expectation step that computes the expected counts with respect to the posterior probability of the topic given document. The other one is the maximization step where the unknown parameters are updated by maximizing the expected complete data log likelihood. For E step, we have to apply Bayes' formula:

$$P(z_k, \mid d_i, w_j) = P(w_j \mid z_k)\, P(z_k \mid d_i) \,/\, \sum_{l=1}^{K} P(w_j \mid z_l)\, P(z_l \mid d_i)$$

In M step, we have to maximize complete data log likelihood E [$\mathcal{L}^c$]. Because the P (di) gn (di) is carried out separately. Thus the left part is by:

$$E[\mathcal{L}^c] = \sum_{i=1}^{L}\sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} P(z_k \mid d_i, w_j) \log [P(w_j \mid z_k)\, P(z_k \mid d_i)]$$

For normalization constraints, the previous one should be augmented by τ_k, ρ_i:

$$H\upsilon = E[\mathcal{L}^c] + \sum_{k=1}^{k} \tau_k \left(1 - \sum_{j=1}^{M} P(w_j \mid z_k)\right) + \sum_{i=1}^{L} \rho_i \left(1 - \sum_{k=1}^{K} P(z_k \mid d_i)\right)$$

To maximize Hυ with associated probability mass functions will make the stationary equations:

$$\sum_{i=1}^{L} n(d_i, w_j)\, P(z_k, \mid d_i, w_j) - \tau_k P(w_j \mid z_k) = 0$$

$1 \le j \le L,\ 1 \le k \le K;$

$$\sum_{j=1}^{M} n(d_i, w_j)\, P(z_k, \mid d_i, w_j) - \rho_i P(z_k \mid d_i) = 0$$

$1 \le j \le N,\ 1 \le k \le K;$

After Lagrange Multipliers eliminated, one contains the re-estimation equations:

$$P(w_j \mid z_k) = \sum_{i=1}^{L} n(d_i, w_j)\, P(z_k, \mid d_i, w_j) \,/\, \sum_{m=1}^{M}\sum_{i=1}^{L} n(d_i, w_m)\, P(z_k, \mid d_i, w_m)$$

$$P(z_k \mid d_i) = \sum_{j=1}^{L} n(d_i, w_j) P(z_k, \mid d_i, w_j) / n(d_i)$$

We iterate these two steps until convergence.

2.4 Measure of language model quality

The most desirable measure of success of a language model is to put the language model into the decoder of machine translation systems and see whether it can improve the BLEU scores. But this is an extremely difficult task special the language model is complex. As an alternative, a statistical language model can be evaluated by how well it predicts a string of symbols W_t, commonly referred to as *test data,* that is generated by the source to be modeled.

Assume we want to compare two language models M_1 and M_2; they assign probabilities $P_{M_1}(W_t)$ and $P_{M_2}(W_t)$, respectively, to the sample test word string W_t The test word string has neither been used nor seen at the parameter estimation step of either model, and we assume that it was generated by the same source that we are trying to model. We consider M_1 to be a better model than M_2 if $P_{M_1}(W_t) > P_{M_2}(W_t)$. The most commonly used quality measure for a given language model M is related to the entropy of the underlying generative source model, and was introduced by IBM researchers under the name of perplexity (PPL) (Jelinek et al, 1977) in 1970s, and the perplexity is defined as

$$PPL(M) = \exp(-1/T \sum_{k=1}^{T} \log[P_M(w_k \mid W_{k-1})]) \quad (3)$$

Where $W_{k-1} = w_1, w_2, ..., w_{k-1}$ denotes the history of w_k and T denotes the length of test data.

Chapter 3

Composite N-gram/PLSA

3.1 Description

We combine N-gram and PLSA together to establish a composite generative language model. In this case, to generate the next word w_{k+1}, it will depend not only on the N-gram history w_{k-n+2}^{k} but also semantic content z_{k+1}. The new parameter in the composite becomes $p(w \mid w_{-n-1}^{-1} z)$. Figure 3 is a graphical representation of the composite N-gram/PLSA language model:

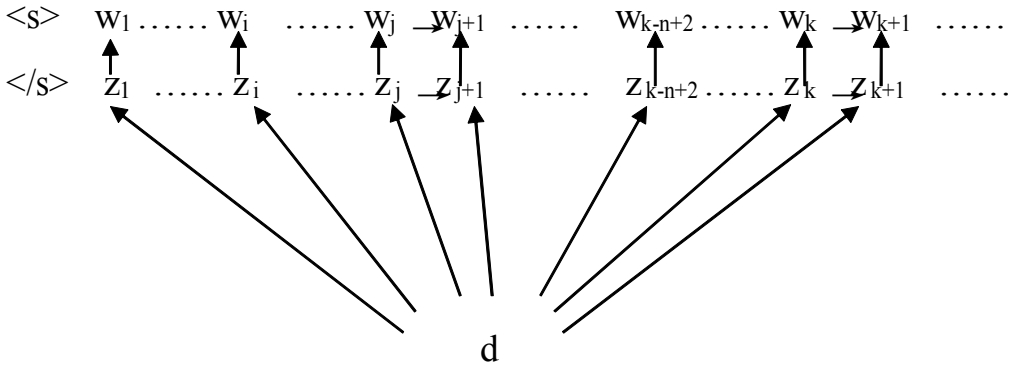

Figure 2 A composite N-gram/PLSA language model that generates w_{k+1} with probability $p(w_{k+1} \mid z_{k+1})$ instead of $p(w_{k+1} \mid w_{k-n+2}^{k})$ and $p(w_{k+1} \mid z_{k+1})$ respectively

The composite N-gram/PLSA language model can be formulated as a chain-table directed dMarkob random field (MRF) where its parameters are constrained by local normalization as the following:

$$\sum_{w \in V} p(w \mid w_{-n+1}^{-1} z) = 1$$

$$\sum_{g \in G} p(z \mid d) = 1$$

3.2 Training Algorithm

For the composite N-gram/PLSA language model und, the likelihood of a training corpus D, a collection of documents, can be written as

$$\mathcal{L} = (\mathcal{D}, \mathrm{p}) = \prod_{d \in D} ((\prod_{l} (\sum_{G^l} P_p(W^l, G^l \mid d)) p(d)))$$

and

$$P_p(W^l, G^l \mid d) = \prod_{z \in G} p(z \mid d)^{\#(z, W^l, G^l, d)} (\prod_{w, w_{-1}, \ldots, w_{-n+1} \in V} p(w \mid w_{-n+1}^{-1} z)^{\#(w_{-n+1}^{-1}, z, W^l, G^l, d)})$$

where #(z,W^l,Z^l,d) is the count of semantic content z in semantic annotation string Z^l of the lth sentence W^l in document d, and #(w_{-n+1}^{-1} ,z, W^l,G^l,d) is the count of N-grams and semantic content z in semantic annotation string Z^l of the lth sentence W^l in document d.

We use maximum likelihood estimation principle to estimate the unknown parameters in the composite N-gram/PLSA language model. Since the semantic content z is hidden, we have to use EM algorithm to perform this task.

For E step, we have to apply Bayes' formula:

$$P(z_k, \mid d_i, w_j, w_{j-1}) = P(w_j \mid z_k, w_{j-1})\, P(z_k \mid d_i) / \sum_{l=1}^{K} P(w_j \mid z_l, w_{j-1})\, P(z_l \mid d_i)$$

In M step, we have to maximize complete data log likelihood $E[\mathcal{L}^c]$. Because the P(di) g n(di) is carried out separately. Thus the left part is by:

$$E[\mathcal{L}^c] = \sum_{i=1}^{L}\sum_{j=1}^{M} n(d_i, w_j) \sum_{k=1}^{K} P(z_k \mid d_i, w_j, w_{j-1}) \log [P(w_j \mid z_k, w_{j-1})\, P(z_k \mid d_i)]$$

For normalization constraints, the previous one should be augmented by τ_k, ρ_i:

$$H\upsilon = E[\mathcal{L}^c] + \sum_{k=1}^{K} \tau_{k,j-1+N}\left(1 - \sum_{j=1}^{M} P(w_j \mid z_k, w_{j-1})\right) + \sum_{i=1}^{L} \rho_i \left(1 - \sum_{k=1}^{K} P(z_k \mid d_i)\right)$$

To maximize $H\upsilon$ with associated probability mass functions will make the stationary equations:

$$\sum_{i=1}^{L} n(d_i, w_j)\, P(z_k \mid d_i, w_j, w_{j-1}) - \tau_k P(w_j \mid z_k, w_{j-1}) = 0$$

$1 \le j \le M,\ 1 \le k \le K;$

$$\sum_{j=1}^{M} n(d_i, w_j)\, P(z_k \mid d_i, w_j, w_{j-1}) - \rho_i P(z_k \mid d_i) = 0$$

$1 \le j \le L,\ 1 \le k \le K;$

After Lagrange Multipliers eliminated, one contains the re-estimation equations:

$$P(w_j \mid z_k, w_{j-1}) = \sum_{i=1}^{L} n(d_i, w_j) P(z_k \mid d_i, w_j, w_{j-1}) / \sum_{m=1}^{M} \sum_{i=1}^{L} n(d_i, w_m) P(z_k \mid d_i, w_m, w_{j-1})$$

$$P(z_k \mid d_i) = \sum_{j=1}^{L} n(d_i, w_j) P(z_k \mid d_i, w_j) / n(d_i)$$

We iterate these two steps until convergence.

3.3 Distributed Architecture

When we train our composite N-gram/PLSA language model on a large scale corpus, both data and model parameters could not be stored in the memory of a single machine. To overcome this problem, we adopt the standard MapReduce paradigm used by Brants et al. (Brants et al. 2007) for N-grams. See Figure 5 for an illustration.

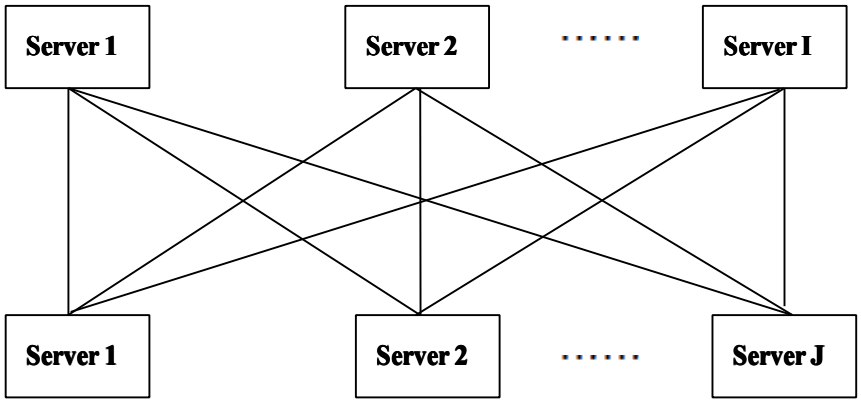

Figure 3.: Distributed Architecture perform the EM algorithm to train composite N-gram/PLSA language model

The corpus is divided and loaded into a number of clients. We run a pure PLSA to extract a list of most likely topics for each document, thus we are able to get the initial counts for $w_{-n+1}^{-1}wz$, thus we finish the Map part, and then the counts for a particular $w_{-n+1}^{-1}wz$ at different clients are summed up and stored in one of the servers by hashing through the word w and its topic z at different clients are summed up and stored in one of the servers,

we then finish the Reduce part. This is the initialization of the EM step.

Each client then calls the servers for parameters to compute the expected count for a particular parameter of $w_{-n+1}^{-1}wz$, thus we finish a Map part, then the expected count of $w_{-n+1}^{-1}wz$ are summed up and stored in one of the servers by hashing through the word w and its topic z, thus we finish the Reduce part. We repeat this procedure until convergence.

If we have access to a large cluster of machines with Hadoop installed that are powerful enough to process a billion tokens level corpus, we just need to specify a map function and a reduce function etc., Hadoop will automatically parallelize and execute programs written in this functional style. Unfortunately, we don't have this kind of resources available. Instead, we have access to a supercomputer at Ohio supercomputer center (OSC) with MPI installed that has more than 1000 core processors usable. Thus we implement our algorithms using C++ under MPI on the supercomputer at OSC, where we have to write C++ codes for Map part and Reduce part, and the MPI is used to take care of message passing, scheduling, synchronization etc. between clients and servers. This involves a fair amount of programming work. Even though our implementation under MPI is not as reliable as under Hadoop, it is more efficient. We use up to 1000 core processors to train the composite language models for 1.3 billion tokens corpus where around 900 core processors are used to store the parameters alone.

3.4 Testing methods

Since a document of the test data is not contained in the original training corpus, to compute the language model probability assignment for word w_{k+1}, we use "fold-in"' heuristic approach similar to the one used in (Hofmann: 2001): the parameters corresponding to $p(z \mid d)$, are re-estimated by maximizing the probability of word subsequence seen so far,

i.e., a pseudo-document $\tilde{d}_k = (W_k, S)$, here S is the set of previous sentences of a document in test data, while holding the other parameters fixed. Wang et al. (Wang et al. 2005b) use online gradient ascent to re-estimate these parameters. We use three methods, *one* online EM and online EM with fixed learning rate use equation below where γ is set to be equal to $\frac{1}{|\tilde{d}_k|+1}$ and a constant 0.2 respectively.

$$p(z \mid d_k) = \gamma \frac{p(w_k \mid w_{k-n+1}^{k-1} z) p(z \mid \tilde{d}_{k-1})}{\sum_{z \in G_d} p(w_k \mid w_{k-n+1}^{k-1} z) p(z \mid \tilde{d}_{k-1})} + (1-\gamma) p(z \mid \tilde{d}_{k-1})$$

The batch EM is the standard EM algorithm where we keep the iterative procedure until convergence. The initial values are set to $\frac{\sum_{d \in D} \#(d) \frac{p(z \mid d)}{\sum_{z_i \in G_d} p(z_i \mid d)}}{|D|}$ where for the topics that are purged, we just plug-in 0 for p (g | d). # (d) is the number of words in document d, $d \in \mathcal{D}$, $|\mathcal{D}| = \sum_d \#(d)$ denotes the size of training corpus which is the total number of words in the entire training corpus.

We find that the perplexity results are sensitive to these three methods and the initial values. For example, for batch EM, if we set initial values to be those obtained by using the pseudo-document up to the previous word $\tilde{d}_{k-1} = (W_{k-1}, S)$ and trained by batch EM, we obtain worse perplexity results. Table 6 gives perplexity results which uses these three methods to re-estimate the parameters of $p(z \mid d)$, where the online EM with fixed learning rate not only has the cheapest computational cost but also leads to highest perplexity reductions.

Chapter 4

Experimental Results

4.1 Experiment Setup

We train our language models using three different training sets: one has 44 million tokens, another has 230 million tokens, and the other has 1.3 billion tokens. An independent test set which has 320k tokens is chosen. The independent check data set used to determine the linear interpolation coefficients has 1.7 million tokens for the 44 million tokens training corpus, 13.7 million tokens for both 230 million and 1.3 billion tokens training corpora. All these data sets are taken from the LDC English Gigaword corpus with non-verbalized punctuation and we remove all punctuation. Table 1 gives the detailed information on how these data sets are chosen from the LDC English Gigaword corpus.

Table 1

LDC English Gigaword corpus is chosen used in this experiment. This table shows a specified corpus denoted by afp, afw, nyt and xin in sections.

	1.3 BILLION TOKENS TRAINING CORPUS
AFP	19940512.0003 ～19961015.0568
AFW	19941111.0001 ～19960414.0652
NYT	19940701.0001 ～19950131.0483
NYT	19950401.0001 ～20040909.0063
XIN	19970901.0001 ～20041125.0119
	230 MILLION TOKENS TRAINING CORPUS
AFP	19940622.0336 ～19961031.0797
AFW	19941111.0001 ～19960419.0765
NYT	19940701.0001 ～19941130.0405
	44 MILLION TOKENS TRAINING CORPUS
AFP	19940601.0001 ～19950721.0137
	13.7 MILLION TOKENS CHECK CORPUS
NYT	19950201.0001 ～19950331.0494
	1.7 MILLION TOKENS CHECK CORPUS
AFP	19940512.0003 ～19940531.0197
	320K TOKENS TEST CORPUS
XIN	20041125.0121 ～20041130.0347

The word vocabulary sizes in all three cases are 60 k, which is open, i.e., all words outside the vocabulary are mapped to the <unk> token. These 60k words are chosen from the most frequently occurred words in 44-millions tokens corpus.

4.2 Results

During training, we have to keep only a small set of topics due to the consideration in both computational time and resource demand. Table 2 shows the perplexity results and computation time of composite N-gram/PLSA language models that are trained on three corpora when the pre-defined number of total topics is 200 but different numbers of most likely topics are kept for each document in PLSA, the rest are pruned. For composite 5-gram/PLSA model trained on 1.3 billion tokens corpus, 400 cores have to be used to keep top 5 most likely topics. For composite trigram/PLSA model trained on 44M tokens corpus, the computation time increases drastically with less than 5% percent perplexity improvement. So in the following experiments, we keep top 5 topics for each document from total 200 topics and all other 195 topics are pruned.

Table 2 Perplexity (ppl) results and time consumed of composite n-gram/PLSA language model trained on three corpora when different numbers of most likely topics are kept for each document in PLSA.

CORPUS	n	# OF TOPICS	PPL	TIME (HOURS)	# OF SERVERS	# OF CLIENTS	# OF TYPES OF wwz
44M	3	5	150	0.5	40	100	120.1M
	3	10	143	1.0	40	100	218.6M
	3	20	143	2.7	80	100	537.8M
	3	50	142	6.3	80	100	1.123B
	3	100	142	11.2	80	100	1.616B
	3	200	141	19.3	80	100	2.280B
230M	4	5	110	25.6	280	100	0.681B
1.3B	5	2	56	26.5	400	100	1.790B

	5	5	54	75.0	400	100	4.391B

Table 3 shows comprehensive perplexity results for a variety of different models such as composite N-gram/PLSA and linear combination of N-gram and PLSA etc., where we use online EM with fixed learning rate to re-estimate the parameters of the topic of test document.

TABLE 3 Perplexity results for various language models on test corpus, where + denotes linear combination, / denotes composite model; n denotes the order of N-gram; the topic nodes are pruned from 200 to 5.

LANGUAGE MODEL	44M n=2		230M n=4	Reduction	1.3B n=5	Reduction
BASELINE N-GRAM(LINEAR)	195		150		77	
n-Gram(KNESER-NEY)	182	6.7%	142	5.3%	—	—
PLSA	651	-233.9%	658	-338.7%	688	-793.5%
n-Gram+PLSA	175	10.3%	135	10.0%	73	5.2%
n-Gram/PLSA	150	23.1%	110	26.7%	54	29.9%

Table 4 shows the perplexity results for composite N-gram/PLSA language models when three methods are used to re-estimate the parameters of the SEMANTIZER of test document, we use superscript 1, 2, and 3 to denote that during testing we use one step online EM, online EM with fixed learning rate and batch EM respectively. The online EM with fixed learning rate gives the best perplexity results as well as the least computation time.

TABLE 4 Perplexity results for composite N-gram/PLSA on test corpus, where + denotes linear combination, / denotes composite model; n is the order of n-gram and superscripts 1, 2, 3 denote using one step online EM, online EM with fixed learning rate and batch EM during testing respectively.

LANGUAGE MODEL	44M Reduction n=3		230M Reduction n=4		1.3B Reduction n=5	
N-GRAM(LINEAR)	195		150		77	
N-Gram/PLSA[1]	157	19.5%	118	21.3%	55	28.6%
N-Gram/PLSA[2]	150	23.1%	110	26.7%	54	29.9%
N-Gram/PLSA[3]	153	21.5%	113	24.7%	55	28.6%

Finally, we conduct experiments where we fix the size of training data and increase the complexity of our language models. Since available resources are limited to prevent us to consider complex language models that are trained on 1.3 billion tokens corpus, we consider complex language models trained on 44 million tokens corpus instead. Table 5 shows the perplexity results. We can see that as we increase the order for N-gram from n=3 to n=4, the composite language models become better and have up to 3% perplexity reductions, however, when we increase the order for N-gram to n=5, the composite language models become worse and slightly overfit the data even

if we use linear interpolation smoothing, and there are no further perplexity reductions.

TABLE 5 Perplexity results for various language models on test corpus, where + denotes linear combination, / denotes composite model; n denotes the order of N-gram; the topic nodes are pruned from 200 to 5.

LANGUAGE MODEL	44M n=3	Reduction	44M n=4	Reduction	44M n=5	Reduction
BASELINE N-GRAM(LINEAR)	195		150		77	
n-Gram(KNESER-NEY)	182	6.7%	142	5.3%	—	—
PLSA	651	-233.9%	658	-338.7%	688	-793.5%
n-Gram+PLSA	175	10.3%	135	10.0%	73	5.2%
n-Gram/PLSA	150	23.1%	110	26.7%	54	29.9%

In our last experiment, we apply our composite language model trained by 1.3B word corpus to re-rank the N-best list in statistical machine translation. We use the same 1000-best list that we used as Zhang et al.(2006). There are 919 sentences consist of this list as a translation model. The trigram language model is used by decoders to train on a 200 million tokens corpus with proper Kneser-Ney smoothing (Jurafsky and Martin 2008). Language model is one of the 11 features of each translation. We re-rank the 1000-best list and exchange our language model with MERT (Och 2003) to optimize the BLEU score (Papineni et al. 2002). We separate data into 10 different sections. Nine of them are selected to tune the weights of the 11 features by MERT and the other one section is used to test the BLEU score.

TABLE 6 BLEU score results for the task of re-ranking the n-best list.

SYSTEM MODEL	BLEU
BASELINE	31.95%
5-GRAM	32.86%
5-GRAM/PLSA[1]	33.44%

From Table 6, we can see that the composite 5-Gram/PLSA language model gives 1.49% BLEU score improvement over the baseline and 0.58% BLEU score improvement over the 5-Gram. The 1000-best list should be variety. There are only 20 or 30 different sentences are in it. When we use N-Gram to re-rank the N-best list, the BLEU score is 33.31%. This performance of machine translation on Hiero is given by Chiang (2007). But, when N-Gram is embedded into one pass decoder in Heiro, the BLEU score will be increasing 37.09% higher. The reason for that is the 1000-best list is not variety. Hopefully, the composite language settled down to one pass decoder would improve the BLEU scores.

Chapter 5

Conclusion

In this thesis, we proposed a generative semantic n-gram language model that combines two well-studied models, one is the probabilistic semantic analysis and the other is the N-gram. We have derived scalable EM algorithm and used corpora with up to 1.3 billion tokens to train our composite semantic N-gram language model on a super computer at Ohio supercomputer center with up to 1000 core processors. We test the performance of the composite PLSA/N-gram language by perplexity and n-best list re-ranking for machine translation. Comparing with simple N-gram, the large scale composite language model has achieved significant perplexity reduction and BLEU score improvement in an n-best list re-ranking task for machine translation.

Bibliography

[1] Bellegarda, J.R., 2000. Exploiting latent semantic information in statistical language modeling. Proc. IEEE 88 (8), 1279–1296.

[2] Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin. 2003. A neural probabilistic language model. Journal of Machine Learning Research, 3:1137-1155.

[3] P. Brown, S. Della Pietra, V. Della Pietra and R. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. Computational Linguistics, 19(2): 263--311.

[4] E. Charniak. 2001. Immediate-head parsing for language models. Proceedings of the 39th Annual Conference on Association of Computational Linguistics, 124-131.

[5] E. Charniak, K. Knight and K. Yamada. 2003. Syntax-based language models for statistical machine translation. MT Summit IX., Intl. Assoc. for Machine Translation.

[6] C. Chelba and F. Jelinek. 2000. Structured language modeling. Computer Speech and Language, 14(4):283-332.

[7] S. Chen and J. Goodman. 1999. An empirical study of smoothing techniques for language modeling. Computer Speech and Language, 13(4): 319-358.

[8] L. Coin, A. Bateman and R. Durbin. 2003. Enhanced protein domain discovery by using language modeling techniques from speech recognition. Proceedings of the

National Academy Sciences, 100(8):4516-4520.

[9] W. Croft and J. Lafferty. 2003. Language Modeling for Information Retrieval. Kluwer International Series on Information Retrieval, 13.

[10] Deerwester, S., Dumais, G. W., Furnas, S. T., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41, 391-407.

[11] Dempster, A. P., Laird, N. M.,& Rubin, D. B.(1977). Maximum likelihood from incomplete data via the EM algorithm. J. Royal Statist. Soc. B, 39, 1-38.

[12] F. Jelinek 1998. Statistical Methods for Speech Recognition. MIT Press.

[13] F. Jelinek and J. Lafferty. 1991. Computation of the probability of initial substring generation by stochastic context-free grammars. Computational Linguistics,17(3):347-360.

[14] F. Jelinek and R. Mercer. 1981. Interpolated estimation of Markov source parameters from sparse data. Pattern Recognition in Practice, E. Gelsema and L. Kanal (Editors), 381-397, North Holland Publishers.

[15]F. Pereira. 2000. Formal grammar and information theory: together again?Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences, 358(1769):1239-1253, April.

[16] Gilula, Z., & Haberman, S. J. (1986) Canonical analysis of contingency tables by maximum likelihood. Journal of the American Statistical Association, 81(395),780-788

[17] B. Roark. 2001. Probabilistic top-down parsing and language modeling. Computational Linguistics, 27(2):249-276.

[18] T. Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. Machine Learning, 42(1):177-196.

[19] Hofmann, T., Puzicha, J., & Jordan, M. I. (1999) Unsupervised learning from dyadic data. In Advances in Neural Information Processing Systems, Vol. 11. MIT Press

[20] M. Johnson and E. Charniak. 2004. A TAC-based noisy channel model of speech pairs. Proceedings of the 42th Annual Conference on Association of Computational Linguistics, 33-39.

[21] Katz, S. M. (1987). Estimation of probabilities for sparse data for the language model component of a speech recognizer. IEEE Transactions on Acoustics, Speech and Signal Processing. 35(3), 400-401

[22] Lee, D. D. & Seung, H. S. (1999) Learning the parts of objects by non-negative matrix factorization. Nature, 401(675), 788-791

[23] K. Mark, M. Miller and U. Grenander. 1996. Constrained stochastic language models, In Image Models and Their Speech Model Cousins, S. Levinson and L. Shepp (Editors), 131-137, Springer-Verlag.

[24] J. Ponte and W. Croft. 1998. A language modeling approach to information retrieval. Proceedings of the ACM SIGIR, 275-281.

[25] R. Rosenfeld. 2000. Incorporating linguistic structure into statistical language models. Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences, 358(1769):1311-1324.

[26] R. Rosenfeld. 2000. Two decades of statistical language modeling: where do we go from here? Proceedings of IEEE, 88 (8):1270-1278.

[27] L. Saul and F. Pereira. 1997. Aggregate and mixed-order Markov models for statistical language processing. Proceedings of the Second Conference on Empirical Methods in Natural Language Processing, 81-89. ACM Press.

[28] Salton, G. &McGill, M. J. (1983) Introduction to Modern Information Retrieval. New York: McGraw-Hill.

[29] Saul, L. & Pereira, F. (1997) Aggregate and mixed-order Markov models for statistical language processing. In Proceedings of the 2nd International Comference on Empirical Methods in Natural Language Processing, pp. 81-89

[30] C. Shannon. 1948. A mathematical theory of communication. Bell System Technical Journal, 27(2):379-423.

[31] J. Turner and E. Charniak. 2005. Supervised and unsupervised learning for sentence compression. Proceedings of the 43th Annual Conference on Association of Computational Linguistics, 290-297.

[32] Wallach, H. M. (2006). Topic modeling: beyond bag-of-words. Proc. Int'l. Conf. on Machine Learning, 977-984.

[33] Witten, I. H. & Bell, T. C. (1991) The zero-frequency problem-estimating the probabilities of novel events in adaptive text compression. IEEE Transactions on Information Theory, 37(4); 1085- 1094

[34] P. Xu and F. Jelinek. 2007. Random forests and the data sparseness problem in language modeling. Computer Speech and Language, 21(1):105-152.